

Location-based Queries and Query Representation in Lifelog Retrieval

Bachelor thesis

Faculty of Science of the University of Basel
Department of Mathematics and Computer Science
Databases and Information Systems Group (DBIS)
<https://dbis.dmi.unibas.ch/>

Examiner: Prof. Dr. Heiko Schuldt
Supervisor: Silvan Heller, MSc.

Sanja Popovic
sanja.popovic@stud.unibas.ch
18-054-593

28.07.2021

Acknowledgments

First of all, I would like to thank Prof. Dr. Heiko Schuldt for the opportunity to write my Bachelor thesis in the Databases and Information Systems Group. I also want to thank my supervisor Silvan Heller for his support throughout this thesis. Last but not least, I am thankful for the willingness of the participants who took part in the evaluation and those who reviewed my thesis.

Abstract

Today's technology allows modern devices to offer high storage capacity that is nowadays more affordable compared to only few years ago. Together with low-cost sensing devices, all kinds of data accumulate over time which poses a challenge to multimedia retrieval. The annual Lifelog Search Challenge (LSC) is a competition which addresses this problem. The competition gathers multimedia retrieval systems which compete against each other in finding a sought image which is described in words. All systems work on the same dataset which stems from a lifelogger who continuously took images over multiple months. One of the systems that participates each year in the LSC is vitrivr, a multi-modal retrieval system. Since image descriptions in LSC tasks often include spatial information, the idea was to extend vitrivr by a new modality, i.e., location-based search. In this thesis, vitrivr is extended by a new location-based query formulation and presentation. The thesis also discusses the applied concepts and evaluates the user experience which finally led to encouraging results.

Table of Contents

Acknowledgments	ii
Abstract	iii
1 Introduction	1
1.1 Multimedia Retrieval	1
1.2 Lifelog Dataset	2
1.3 Goal	2
1.4 Outline	2
2 Related Work	3
2.1 vitrivr	3
2.2 Lifelog Search Challenge Participants 2020	5
2.3 City-Stories	7
3 Location-based Query	8
3.1 Concepts	8
3.1.1 Transformation	8
3.1.2 Distance	8
3.1.3 Correspondence Function and Scoring	9
3.1.4 Result Fusion	11
3.2 User Interface	12
3.2.1 Location-based Query Formulation	12
3.2.2 Location-based Result Presentation	12
3.3 Implementation	14
3.3.1 Map	14
3.3.2 Data Structure	14
3.3.3 Autocomplete	15
3.3.4 Data Import	15
3.3.5 Map Feature Module	16
4 Evaluation	17
4.1 Setup	17
4.2 User Evaluation	17

Table of Contents	v
4.3 Results	18
5 Conclusion	21
5.1 Improvement Ideas of Evaluation Participants	22
5.2 Future Work	23
Bibliography	24
Appendix A User Evaluation	26

1

Introduction

Today's technology is developing fast. The storage capacity in devices grows, becomes more affordable and huge collections of images, videos, audio files and more can be accumulated over time. Searching for a specific item, e.g., an image, inside a large collection is inherently difficult and arises a challenge to multimedia retrieval systems.

1.1 Multimedia Retrieval

Multimedia retrieval is concerned with finding a specific document (e.g., image, video) in a multimedia collection. The question is how the data should be saved, compared and retrieved. A document counts as unstructured data and is rather difficult to retrieve from a collection. Given the description of a document, the task is to propose documents that match the description as closely as possible. In order to find a document in a collection, the properties of the sought document and the properties of the stored ones must be compared. This is inherently difficult because a document can be described in many different ways. Structured data, on the other hand, is easy to retrieve because it has defined domains and can be stored in a relational database.

Some multimedia systems provide various modalities (e.g., Query-by-Example, Query-by-Sketch) to the user so that they can describe the sought document as accurately as possible. The various modalities can be combined into one query. With this, the sought document can be more accurately described. One possible modality would deal with spatial information. Location annotations often come with the metadata of an image. When searching, documents might often be linked by the user to locations where they were captured, which would be a helpful tool for retrieving.

vitivr¹ is a multi-modal multimedia retrieval system. It offers querying for an image or a video by combining different types of query terms (e.g., Query-by-Example, Query-by-Sketch, Query-by-Tag). A query can consist of one to many query terms. Each query term

¹ <https://vitivr.org/>

has one or more feature modules assigned which will each retrieve results that are the most similar regarding the feature. The results will be fused into a final result list and returned to the user.

1.2 Lifelog Dataset

For this thesis, a particular dataset (multimedia collection) is used. The dataset consists of 191'439 images taken by one lifelogger [6]. Thus, it is a large collection of images and perfectly suitable and advantageous for testing, developing and improving multimedia retrieval systems. In addition, metadata for each image is provided. It contains information such as timestamps, activities, location names, GPS coordinates, calories, speed, etc. The purpose of the metadata will be explained in later chapters.

The dataset stems from the annual Lifelog Search Challenge (LSC). In this competition, multimedia retrieval systems compete against each other. Expert and novice users have to solve time-limited tasks on the systems. The team whose system delivers the best results in terms of speed and accuracy wins.

All teams work with the same dataset. The task is to find an image with the help of textual descriptions. Each 30 seconds, one further part of the description is revealed. The following is a task from the LSC in 2018 as shown in [5]:

“I was waiting for the train in Dublin city after walking to the station from a sushi restaurant where I had dinner and beer by candlelight. It was on a Tuesday night and I ate in a restaurant called Yamamori.”

1.3 Goal

The goal of this thesis is to extend the existing vitrivr system by a location-based query term and result presentation where the query term can be formulated on a map and the results are also shown on a map. The new extension to vitrivr has also to be evaluated. Another goal is to contribute to vitrivr’s conference paper for LSC 2021 and for this purpose, the papers of all participating systems at LSC 2020 need to be reviewed.

1.4 Outline

This thesis is structured as follows: Chapter 2 presents different works that embed spatial information. In Chapter 3, my contribution is presented by first explaining the conceptual level, the final user interface and then the implementation. The result thereof is evaluated in chapter 4 and the thesis is finally concluded in chapter 5.

2

Related Work

Section 2.1 explains the vitriv system in more detail. The systems at LSC have different approaches regarding query formulation and result presentation which is interesting to take a look at in Section 2.2. In Section 2.3, a mobile application that uses one part of vitriv is shown.

2.1 vitriv

vitriv is a multi-modal content-based multimedia retrieval system that makes it possible to find an image or a video in a large pool of documents. It consists of three layers: Cottontail-DB, Cineast and vitriv-ng. Together they build a stack where each layer has a particular functionality which is illustrated in Figure 2.1. Cottontail-DB is the storage layer, Cineast is the retrieval engine and vitriv-ng is the web-based user interface [8]. With the help of the vitriv system, a document can be described and similar documents will be retrieved.

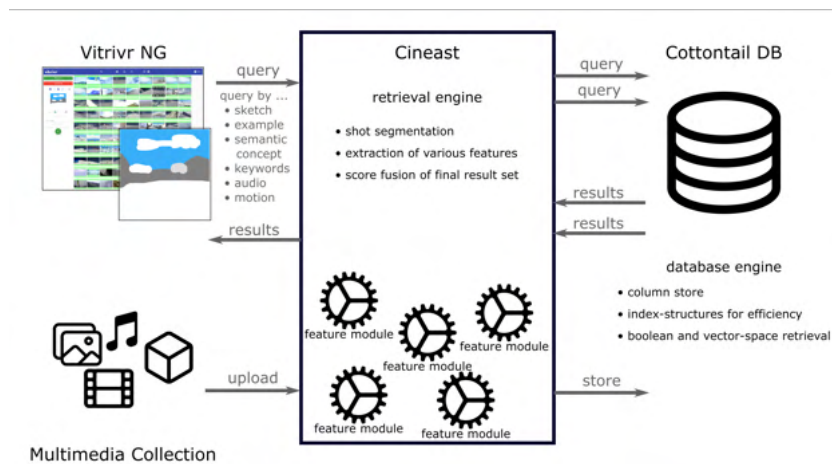


Figure 2.1: The vitriv system and the three components. [9]

vitriv-ng is the user interface where the user can create a query. It offers various combinable modalities to describe a sought image. Figure 2.1 shows a selection of possible query terms

that can be combined into one query, i.e., Query-by-Sketch, Query-by-Example and more. Cineast receives the query via its API. To find documents whose properties are as similar as possible to the sought one's, the query object and the documents in the database must be compared. In Cottontail, in addition to the document itself, its representation as feature vectors is stored as well. In order to compare a query to stored documents, the query must also be transformed to feature vectors. This is done by Cineast's feature modules. Feature vectors are important because they hold information about a document's specific feature such as the global color, average color, spatial information and much more. Depending on the modalities, different feature modules get activated and generate different vectors.

In order to compare feature vectors, a distance measure has to be introduced. Cineast's feature modules have specific distance measures assigned (e.g., *Haversine*, *Euclidean*, *Minkowski*). Since we compare vectors, vitrivr performs comparisons in vector space. For each feature vector that is generated from the query object, a k nearest neighbor (k NN) search in Cottontail-DB is initiated. Cottontail-DB returns results for each vector together with the distance to the original feature vector. Cottontail-DB calculates the distance between the original vector and each returned vector in vector space using the specified distance measure. To express similarity instead of dissimilarity, a correspondence function is applied so that the distance is mapped to the interval $[0,1]$. The similarity value is called the score. A score of zero means no similarity and a score close to one means high similarity between two documents regarding the feature specified by the feature vectors. Cineast scores and fuses the results of each vector generated from the query and returns the final result set to vitrivr-ng. This retrieval process happens in Cineast's *online mode*.

Cineast has also an *offline mode*. This mode is activate when a new document is added to the database. As mentioned before, a representation of a document in form of vectors is generated and saved in the database. It is therewith necessary to generate vectors from a new document which is called feature extraction.

LSC Dataset Import: vitrivr imports the LSC dataset into Cottontail-DB in *offline mode*. vitrivr was initially designed for video retrieval, but the LSC dataset consists of images. Therefore, the images that were taken within a day are bundled into so-called multimedia objects [7]. A multimedia object consists of at least one segment, i.e., an image. With this concept, all images taken within a day are the frames of a video. In the feature extraction process, the feature modules generate vectors for each segment.

2.2 Lifelog Search Challenge Participants 2020

Each team that participates at LSC, writes a paper which explains their system. As part of my thesis, I contributed to the conference paper of vitivr [9] for the LSC 2021. I had the chance to review other systems in the *Related Work* part of the paper.

This chapter introduces some of the approaches of other multimedia retrieval systems that participated at the LSC in 2020. The focus will lie on systems that embedded visualization of spatial context regarding query formulation or result presentation.

LSC is a competition in which the fastest and most precise system wins. Since the systems will be tested by expert and novice users, it is important for the user interface not to be complicated and to provide different methods to describe the sought image. Therefore, the developers focused on embedding various query modalities and informative result presentations in their system.

Three systems provide a visualization of spatial context on a map: Myscéal [13], LifeSeeker [11] and the system of Chu et al. [3].

Myscéal Myscéal enables the user to query for a region by drawing a rectangle on the map as visible in Figure 2.2. In addition to that, specific location names such as *The Sisters Home* and regions such as *London* can be textually queried. This feature is helpful in case the user doesn't know where a specific location or a city is exactly located. If the user enters a normal text query visible on the top of Figure 2.2, the results will be shown on a map by default. After executing a query, hovering over the pins in the map will highlight the respective images to the left of the map and the other way around. The map is displayed as part of the result presentation and can be removed if not needed and again added to the view. Besides drawing on the map, the user can create text queries where information such

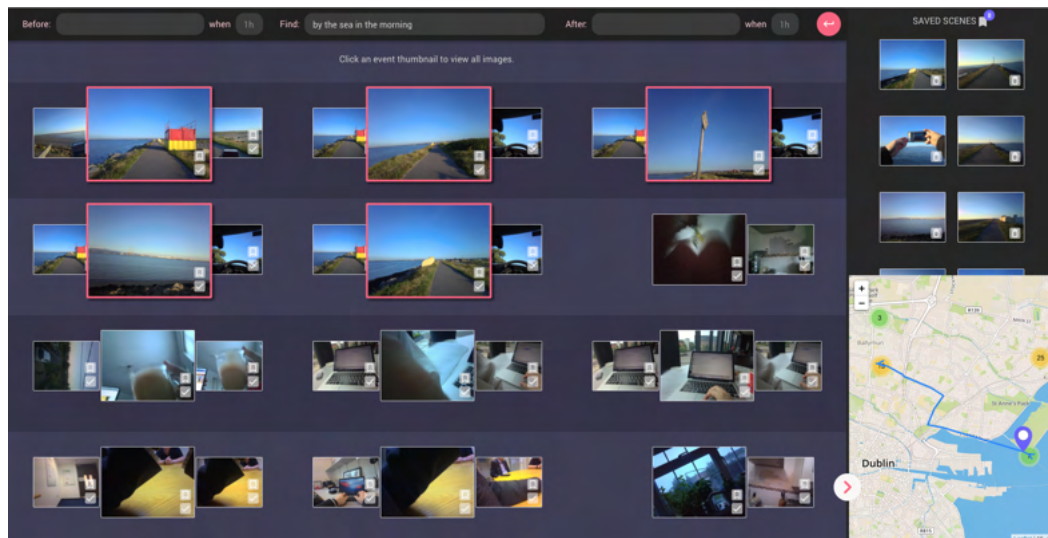


Figure 2.2: The user interface of the Myscéal system. [13]

as date, time, location and objects are automatically extracted by the system. Similar to vitivr, Myscéal also offers expressing temporal queries. It is possible to explain an image

that was taken before and after the sought image and to set the time interval in hours. On the top of Figure 2.2, 2 further text boxes are available where a previous event and a later event can be described. For the result presentation, always three events are bundled. The triple shows the temporal relation, i.e., the first event happened first, the second event is the actual sought one and the third event is the last that happened. An event is a summary of similar images.



Figure 2.3: Location-based graph of LifeSeeker. [11]

highlighted in red as illustrated in Figure 2.3. The user can further select another vertex that a red edge points to such as *Oslo Airport-Norway* and therewith express a spatio-temporal query. Such as Myscéal, LifeSeeker supports freetext queries. After executing the query, the user can choose an image and upon this, further images taken before and after the chosen image appear. With this, temporal information is embedded within the result presentation.

LifeSeeker LifeSeeker visualizes spatio-temporal information with the help of a graph. The graph consists of vertices and directed edges which represent a location and the movement patterns of the lifelogger as presented in Figure 2.3. It has three granularity levels where users can choose between country, location and area [11]. Le et al. explain in [11] that the finest granularity level, i.e., area, shows for example in which room of the lifelogger’s home the image was taken. In this case, the second granularity level, i.e location, would be *Home*. Figure 2.3 shows how the second granularity level looks like. When the user selects a vertex, the outgoing and incoming arrows will be

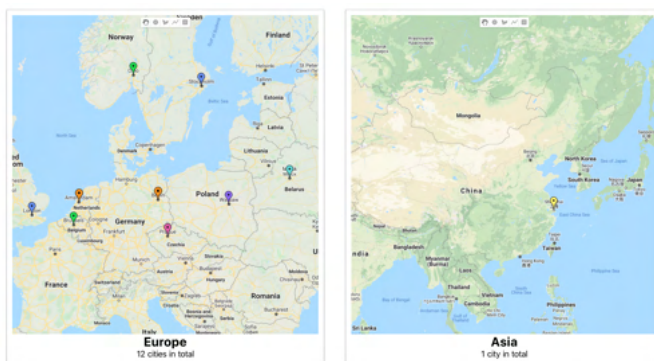


Figure 2.4: Resulting pins after retrieval. [3]

With this, temporal information is embedded within the result presentation.

Chu et al. The system of Chu et al. incorporated a visualization of spatial information only in the result presentation which can be seen in Figure 2.4. The focus lies on displaying locations on the map that have been visited by the lifelogger. Therewith, it is not possible to indicate locations on the map but it is possible to search for a location as keyword. Besides text queries, a location, date and time filter is provided to exclude irrelevant images.

2.3 City-Stories

This section discusses a mobile location-based multimedia retrieval system called *City-Stories* [2]. The mobile application makes it possible to retrieve images. It includes temporal search, spatial search, Query-by-Example and Query-by-Sketch. The query modes can be combined in any way. Moreover, it is possible to take images which are automatically annotated with metadata and saved in the database. The metadata holds the GPS information, i.e., the place and time the image is taken. Figure 2.5 shows the user interface with the results (bottom right), the map (top right), the timeline (top left) and the canvas (bottom left).

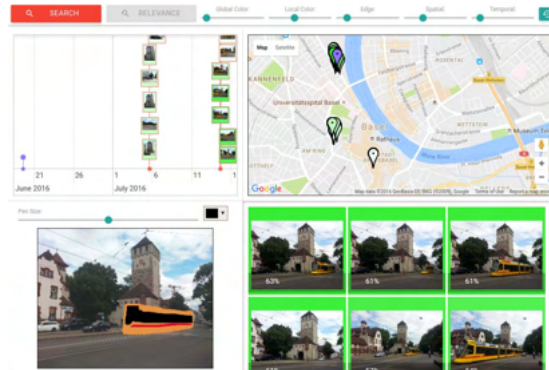


Figure 2.5: The user interface of *City-Stories*.

In the canvas section, the user can sketch an image (Query-by-Sketch) or insert an existing image (Query-by-Example). With the top left section, the user can create a temporal query by adding a purple point in the timeline and the nearest results will be displayed in the section itself and in the section bottom right. To query by spatial information, a purple marker can be created at the desired location and the nearest results will be displayed as markers on the map and on the bottom right as images.

Cineast is used in order to support the different query modalities. As explained in Section 2.1, Cineast has two modes of operation: *offline* and *online mode*.

When a new image is taken, Cineast processes the image by applying feature extraction and saves those features in the database which is the *offline mode*.

The retrieval happens in *online mode* where the query object is transformed into feature vectors by the respective feature modules. A k NN search is initiated for each generated vector. The distance between the generated vector and the retrieved feature vectors is calculated. Two feature modules were added to Cineast, one for spatial similarity and one for temporal similarity. When the user performs a spatial query, a linear correspondence function is applied to the distances and will be further explained in Subsection 3.1.3. In the case of *City-Stories*, if the distance between two vectors is greater than 1000m, the similarity score of zero is assigned. When the user performs a temporal query, the hyperbolic correspondence function is used to assign a score which will be explained more precisely in Subsection 3.1.3. An example in [2] is given which shows how the hyperbolic function scores in *City-Stories*: 100% = 0ms, 90% = 13h20min, 50% = 5 days, 10% = 45 days. 100% means a score of 1 and this would mean that the image was taken at the exact time the user queried for. Images with a small score are more distant in terms of time.

City-Stories is conceptually the same as vitrivr, the only difference being that *City-Stories* works as a mobile app. This is important, as it allows the user to retrieve images conveniently. Using a phone enables taking images and interacting with the environment.

3

Location-based Query

This chapter starts with explaining concepts of multimedia retrieval that are important for this thesis. Then, the final user interface is presented and lastly the implementation is discussed.

3.1 Concepts

This section is based on Section 2.1 which explains the functionality of vitivr. The following four subsections explain the applied concepts needed for the specific location-based retrieval and general retrieval.

3.1.1 Transformation

When the user formulates a map-based query and then executes the search, the GPS data from the user input is collected and then forwarded to Cineast, where the retrieval takes place. The goal is to extract information from the query object to make it comparable to other documents in the vector repository. In our case, a document would be an image and a vector repository the database. The feature module is responsible for retrieval and therefore transforms the query object into vectors. A map-based query holds one to many GPS points that are dividable in latitude and longitude which indicate the location on a sphere. A GPS point could therefore be transformed into a two-dimensional vector in the form of:

$$\vec{v}_i = \begin{bmatrix} latitude_i \\ longitude_i \end{bmatrix} \text{ where } i \text{ indicates } i\text{th vector generated by the feature module.} \quad (3.1)$$

Now, the vectors describe the query and we can compare the generated vectors to others in the vector repository (in our case the database) by calculating the pairwise distances.

3.1.2 Distance

The distance is showing the dissimilarity between two vectors in a vector space. Depending on the feature module, a different distance measure is used. In case of GPS points, the *Haversine* distance is used since the great-circle distance (the shortest path on a sphere) has

to be calculated. Figure 3.1 shows two points A and B on a sphere and the shortest path between them. Equation 3.2 [4] shows how the distance $d_{A,B}$ between the two GPS points is calculated where r is the radius of the sphere.

If we take a look at the *Euclidean* distance defined in Equation 3.3 [12], it is clear that this distance measure is not suitable for a great-circle distance problem since it calculates the distance by considering a straight line between two points. As visible in Figure 3.1, the distance between A and B is not a straight line but a curved one. Equation 3.3 applies to the general case of a n -dimensional vector. In our case, we have a two-dimensional vector and hence $n = 2$.

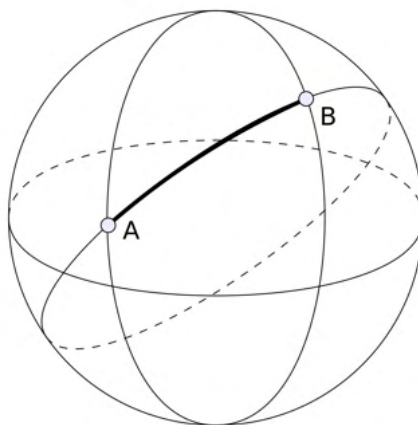


Figure 3.1: The great-circle distance between two points on a sphere. [1].

$$d_{A,B} = 2 \cdot \arcsin \left(\sqrt{\sin^2 \left(\frac{\lambda_B - \lambda_A}{2} \right) + \cos(\lambda_A) \cos(\lambda_B) \sin^2 \left(\frac{\mu_B - \mu_A}{2} \right)} \right) \cdot r \quad (3.2)$$

$$d(\vec{X}, \vec{Y}) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2} \quad (3.3)$$

3.1.3 Correspondence Function and Scoring

Instead of showing the dissimilarity of two vectors, we want to show the similarity by mapping the distance to $[0,1]$. This is done by applying a correspondence function. With the correspondence function, each vector receives a score and therewith a scored list of result can be presented, where a similarity of 0 means no similarity at all and a similarity of 1 means total equality regarding the vectors. Two types of correspondence functions are covered in this thesis: *linear* and *hyperbolic* correspondence functions.

The *linear* correspondence function is defined as $C(x) = \text{clamp}(1 - \frac{x}{x_{max}})$ where x is the calculated distance (3.1.2) and x_{max} is the elimination criteria from which on the score is zero. In Figure 3.2, the *linear* correspondence function is illustrated with different x_{max} values. So for example, if we want to retrieve images taken within a circle, the score of an image will be zero if the location of the image lies outside the circle and therewith the

distance to the circle center is greater than the radius of the circle. In this case, the radius of the circle is the x_{max} value. It is also visible in Figure 3.2 that with a smaller x value, i.e. distance, the score increases. This correspondence function is suitable for location-based image retrieval since the function is monotonically decreasing and zero from the x_{max} value on. From this point on, the $C(x)$ values would be negative but due to the *clamp* function as defined in 3.4 [12], $C(x)$ values smaller than zero will be rounded up to zero.

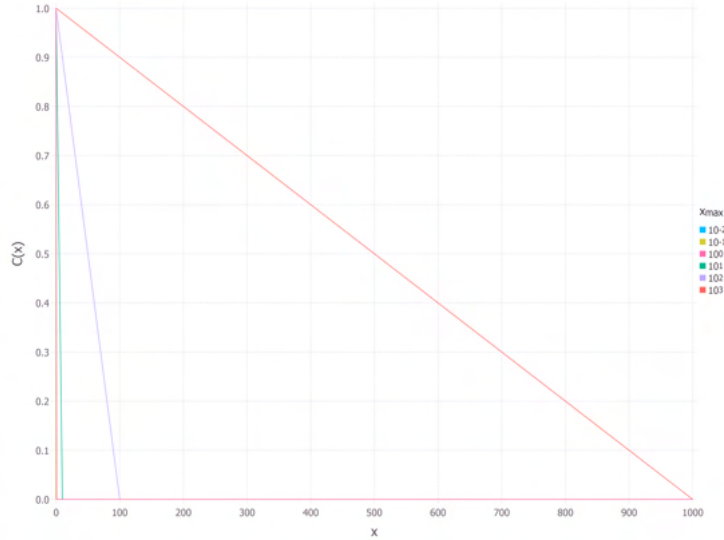


Figure 3.2: Linear correspondence function with different x_{max} values being the limit where within the score is greater zero. [12]

$$clamp(x) = \begin{cases} 0 & x < 0 \\ x & 0 \leq x \leq 1 \\ 1 & x > 1 \end{cases} \quad (3.4)$$

The *hyperbolic* correspondence function $C(x) = clamp(\frac{1}{1+\frac{x}{d}})$ is illustrated in Figure 3.3, where x is the distance and d is the parameter which influences the slope. A smaller d value leads to having less results with a high score. A property of the *hyperbolic* function is that it will never be zero. Another property is that at the x value of $x = d$, the value of the function is 0.5. This correspondence function is suitable for cases where no limit is available, where we can say that from this limit, the score is zero as in the case of a location-based query.

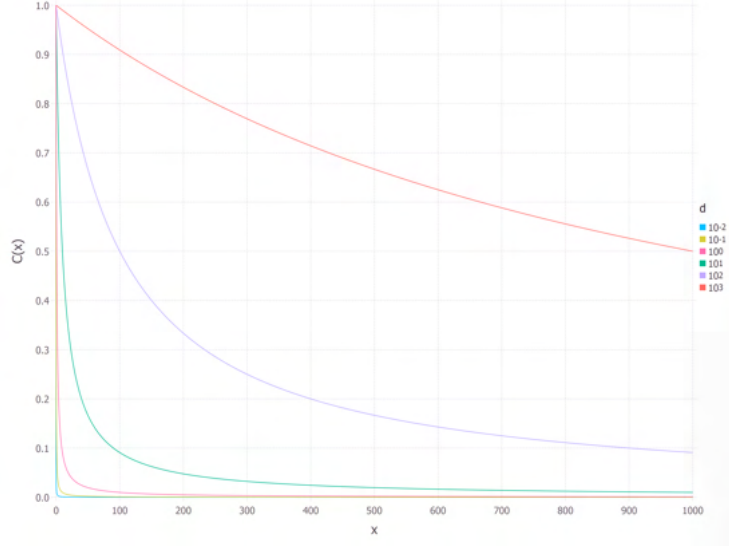


Figure 3.3: Hyperbolic correspondence function with different d values influencing the slope. [12]

3.1.4 Result Fusion

The vectors retrieved by different feature modules have to be combined into a final result list so that each document appears once with a score. To achieve this, the vectors undergo a two-step *weighted score-based late fusion*, as explained in [12].

In the first fusion step, retrieval results of each feature module $f \in F$ that belong to a specific feature category F are fused into one list. Feature modules are bundled into a category based on the information the feature modules work on. An example might be the feature category *GlobalColor* which is composed of different feature modules such as *AverageColor*, *MedianColor*, etc. The feature modules are also differently weighted within the category.

For every distinct element i (i.e., document) that is contained in the set of all elements retrieved in F , the fused score S_i is calculated by first multiplying the weight w_f of the module f with the score $s_{i,f}$ of the element itself in f for each module. Then, the results are summed up. The mathematical formulation can be seen in Equation 3.5 [12]. With this formula, the score of one element over all feature modules within a feature category is calculated. If an element i is not present in one $f \in F$, then $s_{i,f} = 0$ [12].

$$S_i = \sum_{f \in F} w_f \cdot s_{i,f} \quad (3.5)$$

$$0 \leq S_i \leq 1, 0 \leq s_{i,f} \leq 1 \quad \forall i \in I, \forall f \in F \quad (3.6)$$

$$\sum_{f \in F} w_f \stackrel{!}{=} 1 \quad (3.7)$$

The properties in 3.6 [12] show that the final score S_i of each element must be in $[0,1]$. Since the weights w_f are user-defined, the weights first have to be L_1 -normalized as in Equation 3.7 [12] to ensure the properties in 3.6.

Now that we have a list of scored and distinct elements on a feature category level, the second step is to further fuse the elements into a final result list. The process in step one has to be repeated but now instead of feature modules, we are bringing elements from different feature categories together.

3.2 User Interface

This section shows the final user interface. First, it is explained how the location-based query formulation works and how a location-based query can be expressed. Then, the location-based result presentation is described.

3.2.1 Location-based Query Formulation

The user interface deals with capturing the information from the user input. The map query term provides two ways of how a location can be expressed. Images in a vague region or at a specific location can be searched for by drawing a circle or by entering a name such as *Home*.

Figure 3.4 shows how a location-based query can be formulated. The red circle expresses the desire to search for all images taken within that circle. The pin on the map shows up when a specific location, i.e, in this case *The sisters home*, is entered in the search field. For this, an autocomplete list is provided where the user can choose from available locations. The entries in this list are locations that have been visited by the lifeloggers. Images that were taken within the circle around Dublin or at *The sisters home* will be in the result set. Right next to the map, a list of tags is provided which references to the circle and to the pin in the map. Deleting a tag results in deleting a circle or a pin depending on what it references to. In the map, there are four icons that have different functionalities. With the two upper buttons, the user can zoom in and out of the map. The button below makes it possible to draw a circle and with the fourth button with the bin icon, it is possible to delete single circles/pins or everything at once.

3.2.2 Location-based Result Presentation

After executing the query (*Dublin or at The Sisters Home*) shown in Figure 3.4, the scored images will be returned to vitrivr-ng and presented in the new view. The new view as displayed in Figure 3.5 focuses on displaying pins on a map referencing to images taken within a day. The user can use the slider to go through each day (chronologically ordered) and the images taken that day will be displayed below the slider. Only days where the images from the result set originate from will be considered for the slider and the pins will be updated by day change. As example, the visible images in Figure 3.5 are taken on the 11th September of 2016.

In some cases, a movement pattern can be recognized from the arrangement of the pins as in Figure 3.5. The images below the slider are not sorted by score but by timestamp. The first image in the result set is the first image taken that day. It is possible to find

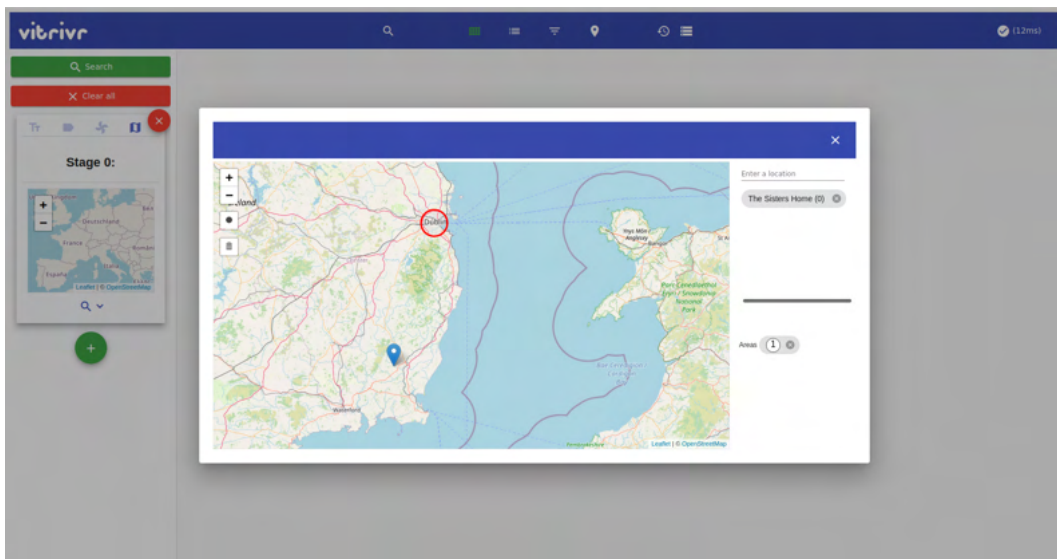


Figure 3.4: Map-based query formulation in vitrivr-ng.

the corresponding pin of an image. When hovering over an image, a small button appears. Clicking on this button results in emphasizing the corresponding pin on the map. It is therewith possible to see where a journey started or ended on the map. It also works the other way around: The user can press a pin and the corresponding image(s) in the result set will be highlighted with red color. Images that have the same GPS coordinates are bundled into one pin. The score is visualized by the green color that frames each image. High color opacity means a higher score and a low one otherwise.

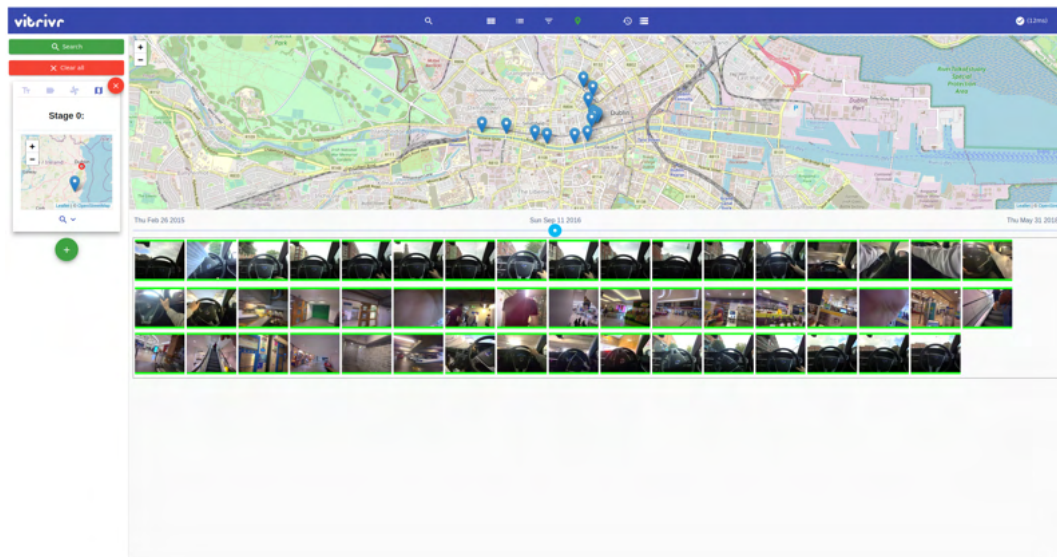


Figure 3.5: Map-based query presentation in vitrivr-ng.

3.3 Implementation

vitivr-ng was developed in the Angular² framework using TypeScript and Cineast was developed in Java. Most of the implementation was done in vitivr-ng. As described in the previous section, the new map query term must support a map which had to be embedded into the existing system and support drawing circles and creating pins. Furthermore, a new result view must have been added to present the results on a map. The following subsections will explain what had to be done to achieve the current user interface shown in 3.2.

3.3.1 Map

For query formulation and result presentation, the open-source library *Leaflet*³ is used. The library provides an editable map. Numerous plugins for editing the map are provided such as creating and deleting different shapes (e.g., circles, pins, rectangles, polylines, polygons), adding popups and much more.

3.3.2 Data Structure

Drawing an area or indicating a specific location produce similar information. Both times GPS coordinates are produced divided in latitude and longitude. Compared to indicating a specific location, surrounding an area additionally generates a distance, i.e., the radius of the circle. With this radius, we say that only images within this radius are accepted as a result.

A very interesting part of the implementation is the data structure that captures the query formulation. When the user enters a specific location (e.g., *The Sisters Home*) or surrounds an area (e.g., circle around Dublin), the GPS coordinates are saved in the same data structure as shown in 3.1. To distinguish between the two, the *type* variable defines if it is a drawn circle or a specific location. In case the user surrounds an area, the circle contains a radius and a latitude and longitude of the center point. These are then saved in the data structure.

Listing 3.1: Circle Data Structure

```
export interface Circle {
  type: string;
  lat: number;
  lon: number;
  rad?: number;
  semantic_name: string;
}
```

When the user selects a specific location from the proposed locations list, then a pin is generated on the map and the latitude and longitude can be extracted from the pin and

² <https://angular.io>

³ <https://leafletjs.com/>

saved in the same data structure *Circle*. That's why the *rad* variable is optional since the pin doesn't have this attribute.

3.3.3 Autocomplete

In order to query for a specific location, a list of locations visited by the lifelogger has to be provided. Figure 3.6 shows how a user can look for available locations. An autocomplete list suggests places that match the letters entered. The proposed location names come together with their GPS coordinates. If the user chooses *Dublin City University (DCU) (0)*, then the GPS data is saved in the data structure shown in Listing 3.1 and a pin with the same GPS coordinates is shown on the map. Images with GPS coordinates not further than 200m from the coordinates of *Dublin City University (DCU) (0)* will be presented after executing the search.

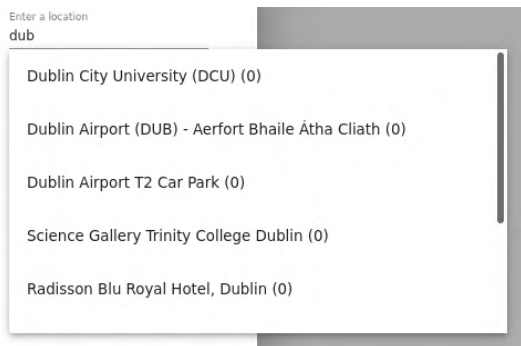


Figure 3.6: The user enters *dub* and multiple suggestions appear.

We have to consider special cases, for instance, that the lifelogger might have moved to another place and kept naming their residence *Home*. Hence, we would have images labelled with *Home* but with different GPS coordinates. The procedure of the import is explained in 3.3.4. That's why each element of the autocomplete list has a number at the end. For example, there are currently four homes (*Home (0)*, *Home (1)*, *Home (2)*, *Home (3)*) available in the dataset, because the GPS data is too widely spread. One reason for the high

frequency of homes might be false annotations in the metadata.

3.3.4 Data Import

To connect to the Cottontail-DB and to create a table, a Python client⁴ is used. There-with a Python script was written which creates a new table *cinest_distinctlocations* in Cottontail-DB and inserts data from a CSV into the database.

To have access to all places the lifelogger went to, a CSV file containing the LSC metadata must be inserted into the database. Since we are only interested in the location names and the GPS coordinates, only these columns will be considered.

Standard	Standard	Standard	Standard	Standard	Standard	Standard	Standard	Standard	Standard	Standard	Standard	Standard
minute_id	utc_time	local_time	timezone	lat	lon	semantic_name	elevation	speed	heart	calories	activity_type	steps
20150223_0000	UTC_2015-02-23_00:00	2015-02-23_00:00	Europe/Dublin	53.3892	-6.15827	Home	NULL	NULL	NULL	1.206200036239624	NULL	NULL
20150223_0001	UTC_2015-02-23_00:01	2015-02-23_00:01	Europe/Dublin	53.3892	-6.15827	Home	NULL	NULL	NULL	1.206200036239624	NULL	NULL

Figure 3.7: Excerpt of the CSV containing the LSC metadata.

Figure 3.7 shows an excerpt of the LSC metadata CSV file which contains 164'160 rows.

⁴ <https://github.com/Spiess/cottontaildb-python-client>

The rows contain information such as the physical activity, the GPS coordinates (latitude and longitude) and more for a given timestamp. To enable autocomplete, three columns had to be considered: *semantic_name*, *lat*, *lon*. The *semantic_name* column contains names of locations and the *lat* and *lon* (abbrev. for latitude and longitude) indicate where this location lies. For example, the two rows in Figure 3.7 show that the lifelogger was at a location called *Home* which has a latitude of 53.3892 and a longitude of -6.15827.

The metadata CSV contains multiple rows with same value for *semantic_name*. For example, the value *Home* appears 71'551 times. The goal is to insert all distinct location names in the *cineast_distinctlocations* table, such that the location names are unique. If all rows which have the same location name have the same GPS coordinates, then there is no problem. The problem occurs when at least two rows with the same value for the *semantic_name* have slightly different GPS coordinates. There are numerous rows which contain the value *Home* for *semantic_name* but different GPS information. Therefore, the closest locations (within three *km*) were bundled and numberings were added to the location names so that the property of uniqueness is established. Locations with the name *Home* (for example) which are too widely spread, will have different numberings.

3.3.5 Map Feature Module

Cineast had to be extended by a new feature module, in order to support the new location-based query. As explained in Section 3.1, feature modules are responsible for transforming the query into feature vector(s). Before sending the data via an API to Cineast, the data saved in Listing 3.1 is parsed into a JSON. In Cineast, a class called *MapQueryContainer* is instantiated when the received data originates from the map query term. This class parses the JSON into one to many *Circle* objects and saves them in a list. The *Circle* class in Cineast corresponds to the *Circle* class in vitivr-ng. Afterwards, the map feature module (*MapSearch* class) accesses the list of *Circle* objects and creates a vector for each element of the list. For each vector, a *kNN* search is executed in Cottontail-DB. Depending on the type of the circle, defined by the *type* variable in Listing 3.1, a maximal distance is set. If it is a circle drawn by the user, then the radius of the circle will be considered and the returned images that lie outside the radius will be scored with zero. If it is a pin created by the user, then the predefined maximal distance setting of 200*m* is considered. Images that lie outside that radius will be scored with zero. The results of all *kNN* searches are collected into a result list such that each image appears only once. If the same image is contained in multiple partial results, then the image with the highest score is taken.

4

Evaluation

This chapter focuses on the evaluation of the location-based query formulation and result presentation. The setup, the goal and the results will be shown.

4.1 Setup

The evaluation was held virtually. Cottontail-DB, Cineast and vitrivr-ng were set up on a server provided by the University of Basel. The participants could access the server with a given IP address and therefore had to activate their VPN client. When the user enters the IP address in the browser, the vitrivr-ng website will be visible, where queries can be formulated. The evaluation was created with Google Forms⁵ and could be therefore filled out online. The evaluation was performed with each participant separately and the participants were kindly asked to share their screen where I could see their interaction with the system. People were invited to the evaluation regardless of their prior knowledge and finally eleven people participated.

4.2 User Evaluation

The main goal is to evaluate the user experience. We want to evaluate how simple, user-friendly and time-efficient querying with the new query formulation and result presentation is. The participants get specific tasks to solve and then get respective questions about the simplicity to answer. Letting the participants solve tasks and really work with the new user interface will result in useful feedback on what they would improve. Since the screen is shared, improvement ideas could also be collected. Participants were allowed to ask questions and we would also point out when a task was totally misunderstood to get the participants on the right track. The whole survey can be found in Appendix A.

The evaluation starts with an introductory question: *Have you already worked with Vitrivr NG?* Since in LSC expert and novice users solve tasks with the system, it would be

⁵ <https://www.google.com/forms/about/>

an advantage if people participate who do and don't know the system. From now on, the questionnaire consists of two parts.

First Part: In the first part, participants had to solve six small tasks and afterwards fill out questions. The tasks were constructed in a way, such that the participants could get to know the map query formulation and result presentation of vitrivr. The even *Likert-Scale* [10] is used and the possible answer options were *Disagreement*, *Slight Disagreement*, *Slight Agreement* and *Agreement*. No *Neutral* option is offered. At the end of part 1, a freetext box is offered in which feedback and possible improvements can be suggested.

Second Part: In the second part, two tasks were given which resemble the tasks at LSC. The goal of the tasks is to find an image that fully meets the image description in the task. The first task is easier and a time limit of four minutes is given whereas the second task is more difficult and has a time limit of eight minutes.

vitrivr-ng in its final form has five query term types. To make sure that the participants will test the extension and not work with other query terms or result views, only two were enabled: *map query term* and *tag query term*. In addition to that, many novice users participated in the evaluation and it might have been overwhelming and difficult to choose from five query terms.

4.3 Results

Just over half of the participants (54.5%) classified themselves as users with prior knowledge that already know vitrivr-ng and the concept of it by affirming the introductory question: *Have you already worked with Vitrivr NG?*

Part 1 Figure 4.1 shows the participants' impressions on how simple it is to work with the map query term and result presentation. All in all, it was mostly classified as simple and easy to do. However, the two slider-related questions have more negative feedback (slight Disagreement).

An interesting part is the freetext section at the end where participants proposed different approaches to improve the user experience of the map query term and result presentation. Participants suggest to add ticks to the slider to show how far the bar has to be moved to reach the next day. It is mentioned that the slider is lagging and it needs some time to load all the images within that day. Participants further suggest to replace the slider with something similar to a calendar, where users can pick a date. Hence, the problem with the ticks and the lagging would be solved.

It is furthermore proposed to make the circle adjustable so that it can be resized and moved after drawing it on the map. Among some participants, drawing the circle was experienced as counterintuitive, since they started to draw the circle from one corner and not from the center. It was also suggested to exchange the circle with other shapes such as rectangles or free-form shapes.

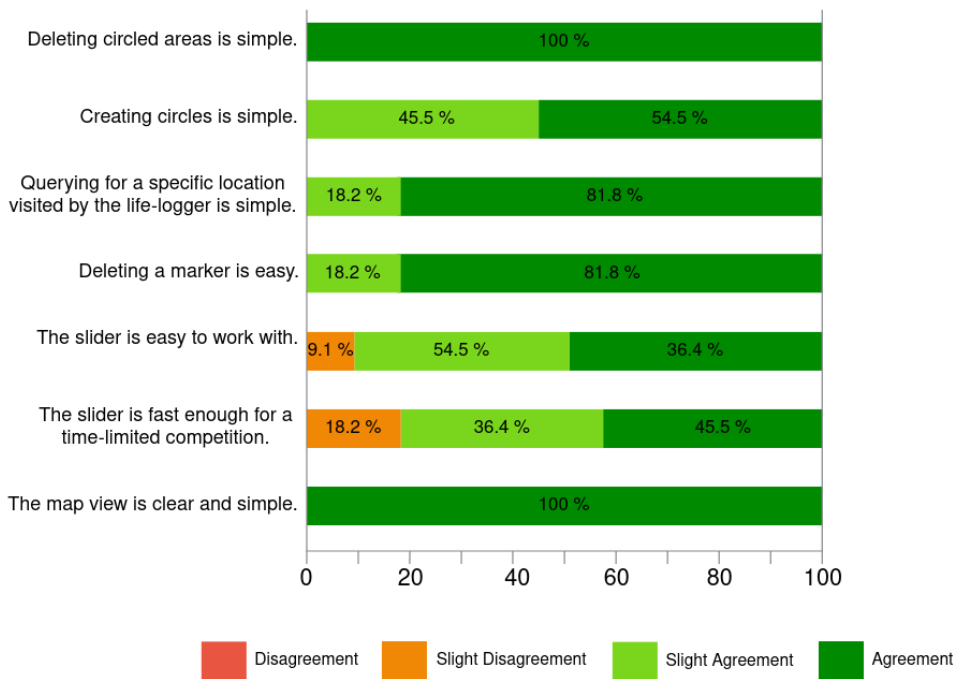


Figure 4.1: Part 1 of the questionnaire with the questions and the ratings.

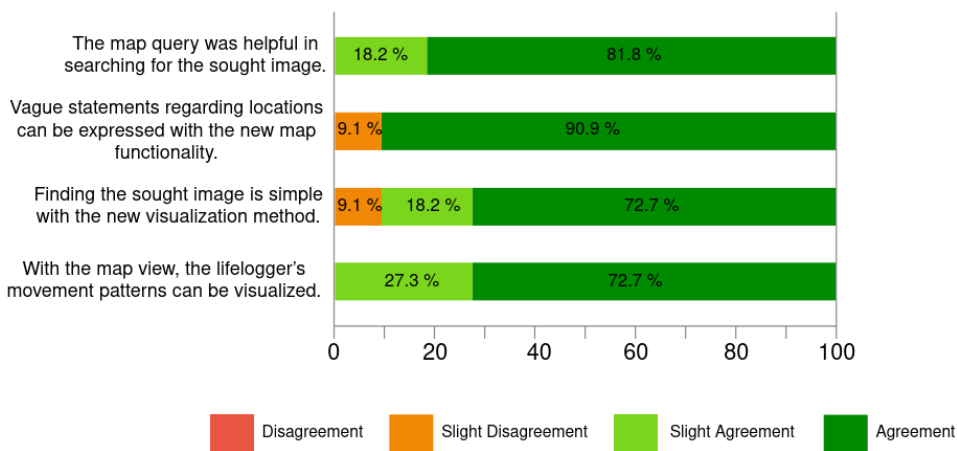


Figure 4.2: Part 2 of the questionnaire with the questions and the ratings.

Part 2 In part two, the usability seems to be high. Many participants experienced the map query term and result presentation helpful when solving the tasks. The participants' opinion about how simple it is to find the sought image with the new result presentation is distinct.

In the freetext feedback section, participants commented that it would be more helpful, if the highlighted images belonging to a selected pin were more noticeable. The red highlighted images are hard to find in the images pool.

Another suggestion is that when the user clicks on a pin, a popup opens above the pin with

the thumbnail of the image that was taken there. In case a pin references to multiple images, the image with the highest score could be taken.

It was further proposed to enable searching for cities and countries. This would speed up the map query term process if the user doesn't know where a location exactly is. An interesting suggestion is to draw a path between single markers to emphasize the lifelogger's movement pattern.

Two participants tried to solve task two with two temporal query containers. This was rather impractical because the time between two query containers is only adjustable in seconds. Longer distances would require minutes or hours.

5

Conclusion

One goal of this thesis was to acquire knowledge about the concepts of multimedia retrieval and to extend vitivr so that formulating spatial queries on a map is possible and that the result presentation is also map-based. This goal was successfully accomplished by providing a location-based query formulation and location-based result presentation. With the new query term, the user can now search for images in vague areas by directly drawing circles on the map. The user can also search for images that were taken at specific places such as *Home, Work, etc.* To enable this, a Python script was written which processes and imports the metadata from the lifelog dataset into Cottontail-DB. The locations of the images in the result set are visualized on a map using pins. A slider is provided where the user can go through different dates. The movement pattern of the lifelogger is visible from the pins on the map. Tools were added to help the user reference between images and pins.

To enable the query formulation and result presentation, work had to be done on Cineast. Cineast was extended by a new feature module which handles the input of the map query and retrieves images that are spatially close by evaluating distances and applying a correspondence function.

Another goal was to obtain an overview over other multimedia retrieval systems and to contribute to the conference paper of vitivr [9] for the LSC 2021 which was successfully submitted. Background knowledge was acquired by reading and summarizing all of the LSC 2020 papers. Since this was the first task in this thesis, many ideas for possible query formulations and presentations could be collected.

Moreover, this thesis provides insights into different multimedia retrieval systems and their approaches. The vitivr system which was extended is explained in more detail by presenting the functionality and the applied multimedia concepts. Afterwards, the required multimedia concepts to realize a map-based query formulation and presentation are discussed. The final user interface is illustrated, the most important parts of the implementation are explained and the applied theoretical concepts are presented. The evaluation shows how the extended user interface is rated among novice and expert participants. The results were remarkably positive and lots of helpful improvement ideas could be collected.

5.1 Improvement Ideas of Evaluation Participants

The map-based query formulation and presentation proved to be helpful, but there are still some shortcomings that should be remedied.

Temporal Query Container: Currently, the temporal query containers only allows to set the time interval in seconds. In case of spatio-temporal queries, it would be necessary to extend the time unit to minutes and hours. If we search for images where the lifelogger moved from one location to another location and the travel took him multiple hours, it is rather impractical to adjust the time in seconds.

Shape: One participant suggested exchanging the circle shape with a rectangle or a polygon shape. This would be a helpful improvement if the user wants to surround more stretched areas, although multiple circles could also be used for this purpose.

Slider: The slider in the result presentation has to be replaced. Since the LSC dataset is large, the loading takes longer. The problem is, that the slider is loading each day between the start till the date the slider is moved to. The amount of images loading should be restricted and only further loaded, if the user scrolls down to search for more images in that day. It is also criticized that the slider has no ticks or labels. The width of the slider stays the same even if two or ten dates are available. The jumps become bigger and this confused the users.

It is suggested to replace the slider with a calendar. This approach would be good but the user would have to go through each month and the day coverage is rather sparse. Another idea would be to place the dates below the map as tags. The user could select one tag and images taken that day will appear.

Image Highlight: In the result presentation, all images per day are shown and the day is adjustable. Taking a look at an example where the user chooses a date where lots of images were taken. If they select a pin on the map and then want to find the red highlighted image in the result set, they might loose the overview. This case occurred in the evaluation and it is suggested to intensify the highlight. One idea would be to immediately let the container, where the images of the day are located in, move down to the image after selecting a pin. Another idea would be to size up the highlighted image(s).

General Location Search: It results from the evaluation that the participants missed a feature where they could search for general locations similar to Google Maps⁶. The participants were searching for a long time for the location *Donegal Bay* in task two (part two of the evaluation).

⁶ <https://www.google.ch/maps/>

5.2 Future Work

The current location-based query formulation and presentation can be conceptually further extended and used with different types of multimedia data.

Sequential Queries: The current map-based query is expandable to a sequential query. For example, the user could look for images that were taken when the lifelogger was going from home to work. The result set would finally contain images that were taken on the way. The temporal aspect should also be taken into account. The result set then only includes images where the lifelogger moved in the right direction, i.e., from home to work and not vice versa. A challenge would also be to visualize the spatio-temporal property of images on a map more clearly.

Video Retrieval: The map-based query formulation and result presentation might be a helpful modality for the Video Browser Showdown (VBS). VBS is similar to LSC but the dataset consists of videos. Since temporal queries are important for video retrieval, the previously mentioned sequential query would also be helpful in finding the sought video.

Bibliography

- [1] Orthodrome. <https://de.wikipedia.org/wiki/Orthodrome#Streckenberechnung>. Accessed: 2021-07-23.
- [2] Lukas Beck and Heiko Schuldt. City-Stories: A Spatio-Temporal Mobile Multimedia Search System. In *IEEE International Symposium on Multimedia (ISM)*, 2016.
- [3] Tai-Te Chu, Chia-Chun Chang, An-Zi Yen, Hen-Hsen Huang, and Hsin-Hsi Chen. Multimodal Retrieval through Relations between Subjects and Objects in Lifelog Images. In *Proceedings of the Third Annual Workshop on the Lifelog Search Challenge (LSC'20)*. ACM, 2020.
- [4] Kenneth Gade. A non-singular horizontal position representation. *Journal of Navigation*, 63(3):395–417, 2010. doi: 10.1017/S0373463309990415.
- [5] Cathal Gurrin, Klaus Schoeffmann, Hideo Joho, Andreas Leibetseder, Liting Zhou, Aaron Duane, Duc-Tien Dang-Nguyen, Michael Riegler, Luca Piras, Minh-Triet Tran, Jakub Lokoč, and Wolfgang Hürst. Comparing Approaches to Interactive Lifelog Search at the Lifelog Search Challenge (LSC2018). *ITE Transactions on Media Technology and Applications*, 7(2), 2019.
- [6] Cathal Gurrin, Tu-Khiem Le, Van-Tu Ninh, Duc-Tien Dang-Nguyen, Björn Þór Jónsson, Jakub Lokoč, Wolfgang Hürst, Minh-Triet Tran, and Klaus Schoeffmann. An Introduction to the Third Annual Lifelog Search Challenge, LSC'20. In *International Conference on Multimedia Retrieval*. ACM, 2020.
- [7] Silvan Heller, Mahnaz Amiri Parian, Ralph Gasser, Loris Sauter, and Heiko Schuldt. Interactive Lifelog Retrieval with vitrivr. In *Proceedings of the Third Annual Workshop on Lifelog Search Challenge*, 2020.
- [8] Silvan Heller, Loris Sauter, Heiko Schuldt, and Luca Rossetto. Multi-Stage Queries and Temporal Scoring in Vitrivr. In *2020 IEEE International Conference on Multimedia Expo Workshops (ICMEW)*, pages 1–5, 2020. doi: 10.1109/ICMEW46912.2020.9105954.
- [9] Silvan Heller, Ralph Gasser, Mahnaz Parian-Scherb, Sanja Popovic, Luca Rossetto, Loris Sauter, Florian Spiess, and Heiko Schuldt. Interactive Multimodal Lifelog Retrieval with vitrivr at LSC 2021. In *Proceedings of the 4th Annual Lifelog Search Challenge (LSC'21)*, 2021.

-
- [10] Ankur Joshi, Saket Kale, Satish Chandel, and D Kumar Pal. Likert scale: Explored and Explained. *British Journal of Applied Science & Technology*, 7(4):396, 2015.
- [11] Tu-Khiem Le, Van-Tu Ninh, Minh-Triet Tran, Thanh-An Nguyen, Hai-Dang Nguyen, Liting Zhou, Graham Healy, and Cathal Gurrin. LifeSeeker 2.0 : Interactive Lifelog Search Engine at LSC 2020. In *Proceedings of the Third Annual Workshop on the Lifelog Search Challenge (LSC '20)*. ACM, 2020.
- [12] Luca Rosetto. *Multi-Modal Video Retrieval*. PhD dissertation, Natural Science Faculty of the University of Basel, 2018.
- [13] Ly-Duyen Tran, Manh-Duy Nguyen, Nguyen Thanh Binh, Hyowon Lee, and Cathal Gurrin. Myscéal: An Experimental Interactive Lifelog Retrieval System for LSC'20. In *Proceedings of the Third Annual Workshop on Lifelog Search Challenge*, 2020.

A

User Evaluation

Evaluation BSc Thesis

1. Have you already worked with Vitivr NG? *

Yes

No

Part 1

The goal is to try out the new map query functionality and therefore please follow the next six instructions given below.

1. Create a map query term and indicate an area surrounding London.
2. Delete the first request for London by deleting the circle.
3. Look for pictures taken in the Dublin city university (DCU) and execute the search.
4. Change to the map view (pin icon at the middle, top edge) and navigate to the 4th of March 2015.
5. Delete all queries and look for a cafe called "The Helix".
6. Delete the marker pointing to the cafe "The Helix".

2. Deleting circled areas is simple. *

	1	2	3	4	
Disagreement	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Agreement

3. Creating circles is simple. *

	1	2	3	4	
Disagreement	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Agreement

Evaluation BSc Thesis

4. Querying for a specific location visited by the life-logger is simple. *

	1	2	3	4	
Disagreement	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Agreement

5. Deleting a marker is easy. *

	1	2	3	4	
Disagreement	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Agreement

6. The slider is easy to work with. *

	1	2	3	4	
Disagreement	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Agreement

7. The slider is fast enough for a time-limited competition. *

	1	2	3	4	
Disagreement	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Agreement

Evaluation BSc Thesis

8. The map view is clear and simple. *

	1	2	3	4	
Disagreement	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Agreement

9. Possible improvements:

Part 2

In part two, you are given two tasks. Please solve the tasks and sub-tasks chronologically and afterwards answer the questions. Since we want to evaluate the new map feature, please use the map view for the result presentation.

Task 1 (max. 4 min for the task!)

- I was at my sisters home.
- It was 2016.
- We celebrated with a cake.
- The number 50 is written on the cake
- There were 4 people in the picture, two of them in the background.

Task 2 (max. 8 min for the task!)

- That day I did a car ride and I was driving.
- The destination was a place somewhere at the coast in the Donegal Bay (in the northwest of Ireland).
- The car ride started in Dublin.
- The trip took place on the 17th of September.
- We're looking for the image where my friend was sitting in the light green grass. That was close to the end of the route.

10. The map query was helpful in searching for the sought image. *

	1	2	3	4	
Disagreement	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Agreement

11. I found the sought image in task 1. *

	1	2	
Disagreement	<input type="radio"/>	<input type="radio"/>	Agreement

12. I found the sought image in task 2. *

	1	2	
Disagreement	<input type="radio"/>	<input type="radio"/>	Agreement

13. Vague statements regarding locations can be expressed with the new map functionality. *

	1	2	3	4	
Disagreement	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Agreement

Evaluation BSc Thesis

14. Finding the sought image is simple with the new visualization method. *

	1	2	3	4	
Disagreement	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Agreement

15. With the map view, the lifelogger's movement patterns can be visualized. *

	1	2	3	4	
Disagreement	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Agreement

16. Possible improvements:

Declaration on Scientific Integrity

Erklärung zur wissenschaftlichen Redlichkeit

includes Declaration on Plagiarism and Fraud
beinhaltet Erklärung zu Plagiat und Betrug

Author — Autor

Sanja Popovic

Matriculation number — Matrikelnummer

18-054-593

Title of work — Titel der Arbeit

Location-based Queries and Query Representation in Lifelog Retrieval

Type of work — Typ der Arbeit

Bachelor thesis

Declaration — Erklärung

I hereby declare that this submission is my own work and that I have fully acknowledged the assistance received in completing this work and that it contains no material that has not been formally acknowledged. I have mentioned all source materials used and have cited these in accordance with recognised scientific rules.

Hiermit erkläre ich, dass mir bei der Abfassung dieser Arbeit nur die darin angegebene Hilfe zuteil wurde und dass ich sie nur mit den in der Arbeit angegebenen Hilfsmitteln verfasst habe. Ich habe sämtliche verwendeten Quellen erwähnt und gemäss anerkannten wissenschaftlichen Regeln zitiert.

Basel, 28.07.2021



Signature — Unterschrift